CS-308-2014 Final Report

FreeRTOS based Greenhouse Manager

TU-3

[A]lliance

Ankush Das, 100050042

Vivek Paharia, 100050044

Pulkit Piyush, 100050038

Nilesh Jagnik, 100050040

Table of Contents

1. Introduction..... 3
2. Problem Statement..... 3
3. Requirements..... 4
3.1 Functional Requirements..... 4
3.2 Non-Functional Requirements..... 4
3.3 Hardware Requirements..... 4
3.4 Software Requirements..... 5
4. System Design..... 5
5. Working of the System and Test results..... 6
5.1 Graphical User Interface..... 6
5.2 Communication between Robot and Greenhouse Manager..... 7
5.3 FreeRTOS based Greenhouse manager..... 7
5.3.1 Sensor Tasks..... 9
5.3.2 Timed Tasks..... 9
5.3.3 Implementation..... 9
5.4 Greenhouse Sensors and Actuators..... 10
5.5 Test Results..... 10
6. Discussion of System..... 11
7. Future Work..... 11
8. Conclusions..... 12
9. References..... 12

1. Introduction

Monitoring and control of greenhouse environment play an important role in greenhouse production and management. To monitor the greenhouse environment parameters effectively, it is necessary to design a measurement and control system. The system developed by us provides the aforementioned functionality. We have designed a simple, easy to install, microcontroller based circuit to monitor and record the values of temperature, humidity, soil moisture and sunlight of the natural environment that are continuously modified and then controlled in order to optimize them to achieve maximum plant growth and yield.

To ensure scalability, we have made use of FreeRTOS, a real-time operating system developed by Real Time Engineers Ltd. A real time operating system is designed to serve real-time application requests. In other words, the application deadlines are deterministically met on such an operating system. We have developed a greenhouse controller based on FreeRTOS. This is the first time a CS308 team is working on FreeRTOS, and thus we intend to make a spoken tutorial available along with other material to help future teams to develop on FreeRTOS.

We have also developed a Java-based desktop application which communicates with the Firebird 5 through XBee radios. This module helps a user to monitor and control the greenhouse manager from any mobile device that has been connected to a XBee radio. It also provides a manual override functionality which gives the control entirely to the user which can be useful in case of a malfunction or emergency.

We have developed a prototype model of a greenhouse in which to which several sensors and actuators are put in contact with the plants. In this way, we have shown that the Greenhouse Manager is successfully working and with some hardware modifications is ready to be deployed to the Greenhouse.

2. Problem Statement

In its current form, the Greenhouse at IIT Bombay has automated periodic irrigation, but temperature and light control is done manually. The problem statement that we are working on is to ensure complete automation of all the functions inside a greenhouse so that very little manual work is required and

all functions can be controlled and monitored by one person using a single interface. This also requires the model to be scalable so that the system installed on a single processor can control large number of greenhouses and agricultural activities. All of the above has been done in an attempt to move towards a futuristic greenhouse model where plants will be grown by programmed machines. A wireless communication technology was needed to achieve the communication between a user and the Greenhouse Manager. We have used XBee for this purpose. We have made a system ready-to-deploy with some configurations that need to be changed according to the hardware. Also, since some functions like light curtains are not automated in the greenhouse yet, our model provides support for any type of modification that may be done to the greenhouse in the coming years. All we are doing is providing power to a actuator based on the reading of the sensors. The actual actuator hardware does not have to be fixed and thus can be one thing now and quite the other after a few years.

3. Requirements

3.1 Functional Requirements

1. Scalability: Our monitor and control system is designed such that it can be used in very big greenhouses. There large of racks of crop are present and each rack has its own set of sensors and actuators. Each such set of sensors is a module in our system. Our system is designed such that each module has finite response time on any change in environment conditions of that module.
2. Modeling of Greenhouse: To test our monitor and control system we need to have small and movable greenhouse. Therefore, we are modeling the greenhouse at small-scale. Cooler, dripping pipes & water pump and series of LEDs will be used in our greenhouse. Our greenhouse is 1x3x5 feet trough with plastic sheets to cover and soil to grow.
3. Cost Effectiveness: In its current form, our system requires very little monetary input and provides a high output. This is because all sensors and actuators required for the system cost very low and even the cost of development of circuit is low. Hence, economically our system is highly effective.

3.2 Non-Functional Requirements

There are some non-functional limitations to our project. We list them below :-

1. We have used LM35 as our temperature sensor. This sensor has a scale factor of 0.01 V/ºC. This does not allow us to accurately predict small changes in the temperature. Also, the range of temperature that the LM35 can handle is 0ºC - 100ºC. So, in extreme temperature conditions, the temperature sensors may not work properly. Additionally, the LM35 sensors wear out over time, and hence, they need to be replaced from time to time to maintain reliability of the circuit.
2. We have used LDR (Light Dependent Resistor) as the light sensor. The resistance across an LDR decreases with increasing incident light intensity. We are operating the LDRs at an output voltage of 5V. Under these conditions, the voltage drop across the resistor varies from 0V-4V. Hence, the light intensity that is displayed has a maximum and a minimum value. So, the thresholds have to be set accordingly. Again, if the light intensity increases beyond the maximum limit, this will not be sensed by the LDR, which might create problems in regions with ambient light. Also,

we switch on the LEDs whenever the light drops below the threshold, and this will be sensed by the LDR. Hence, the LDRs should be placed in a region where the light from LEDs does not reach.

3.3 Hardware Requirements

1. Firebird 5 - Contains the ARM7 processor on which the code is loaded
2. Model Greenhouse - A prototype model using which the project has been tested
3. Sensors - Temperature & Light sensors to keep track of current environment conditions
4. Actuators - Fan, LEDs and other actuators that help bring the environment to a favourable state
5. Circuit - Consists of a PCB board through which the sensors are connected to the Firebird by the means of a circuit which is soldered on the PCB along with resistors, pins, connectors, etc.

3.4 Software Requirements

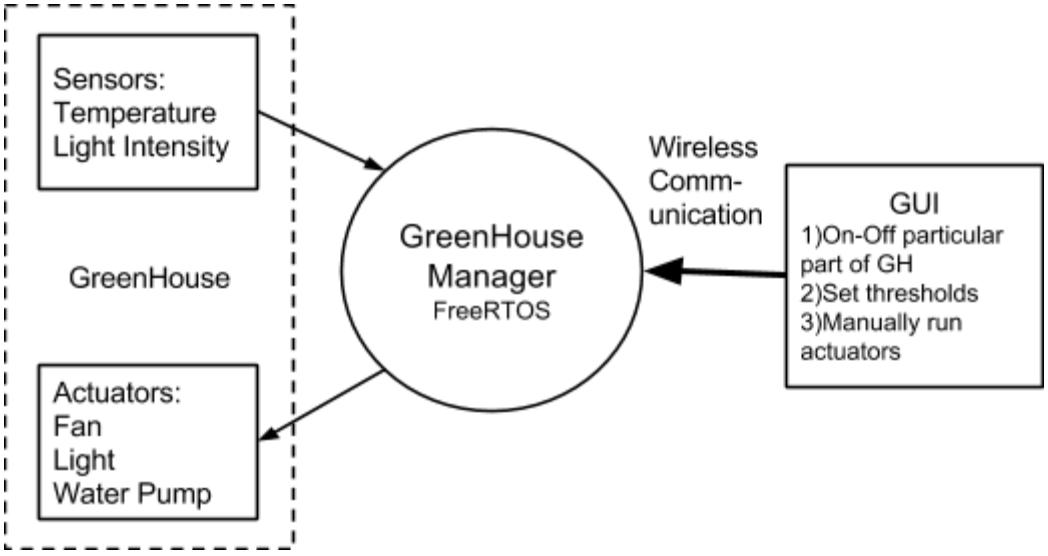
1. Keil uVision 4 - IDE for editing and compiling code
2. FreeRTOS V8 - A real time operating system that supports various architectures
3. Flashmagic - For burning code onto the Firebird 5 using on-chip bootloader
4. Java Simple Serial Connector - To establish serial connection between Firebird and GUI
5. Netbeans - IDE for Java Development using which the GUI was developed
6. Java JRE - The runtime environment is needed to run the Graphical User Interface

4. System Design

The system consists of various components which are as described follows:

1. Graphical User Interfaces
2. Communication Protocol between the Robot and GUI
3. FreeRTOS based GreenHouse manager
4. Greenhouse prototype: Sensors and Actuators

Following diagram shows the complete architecture of the system.



GUI provide easy and convenient way to control the Greenhouse manager. All the instructions give through the GUI are transmitted wirelessly to the bot where Greenhouse manager is running. Greenhouse manager is developed on the FreeRTOS. FreeRTOS tasks are used to run the code in parallel. Along with this functionality FreeRTOS abstract away the scheduling of this tasks. Additionally, FreeRTOS is provides semaphores. Semaphore are used to block the non-functioning tasks to avoid busy waits.

Greenhouse manager also interacts with sensors and actuators. We have modeled a prototype greenhouse in this project. We have 1)temperature sensor(LM35) 2)light sensor(LDR) as sensors and 1)water pump 2)fan 3)LED as actuators. We have prototyped the greenhouse in a cuboid of 36'' x 12'' x 60'' where trough with plant growing in it is kept. These sensors and actuators are installed in this prototype.

5. Working of the System and Test results

5.1 Graphical User Interface

The GUI we provide for controlling the Greenhouse Manager is written in java. Java does not come with pre-defined libraries for serial port communication. So, we have used JSSC, which provides basic serial port IO communication for Java. JSSC internally uses c-based code and uses JNI to port the c-based code to Java. The GUI handles three major functions, besides some utility features. The main Features of the GUI are as follows:-

1. With the GUI, we can manage a part of the GreenHouse. A particular part of greenhouse is assigned a particular sensor set. Through the GUI, we can on-off a particular sensor set via toggle buttons.
2. GUI also allows setting the thresholds for various sensors that are present in our greenhouse. We can set a threshold for any sensor in the greenhouse through the GUI. If the value read from the sensor crosses the threshold, the actuator task for that sensor is switched on.

3. In case of an emergency or if the user is dissatisfied with the performance of our greenhouse, our GUI allows the user to manually override the actuators. With the GUI, users can start the actuators for a fixed time(automatic managing of actuator is disabled for that time).
4. Utility based features provided by the GUI are as follows:-
    a. X-CTU like terminal showing all the data in the serial port communication between the GUI and the bot.
    b. Setting/Loading default values of all the thresholds of the sensors.

5.2 Communication Protocol between the Firebird and GUI

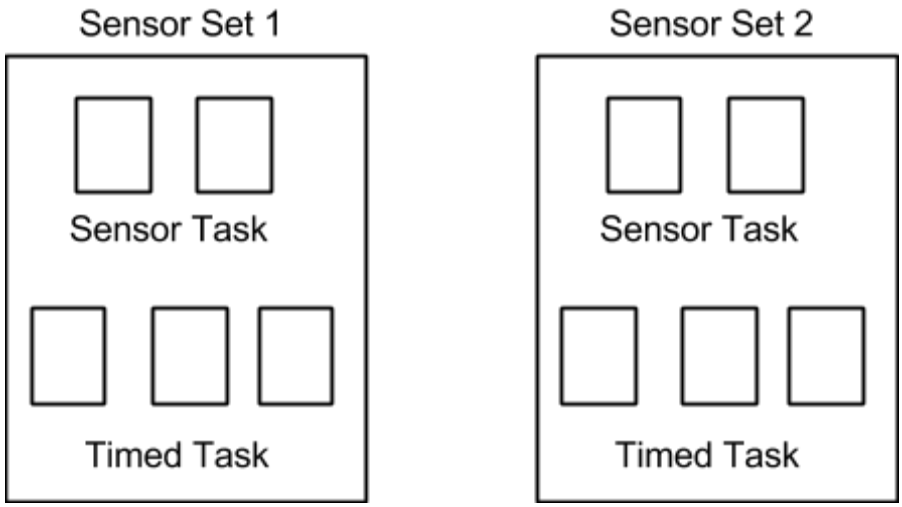
Whenever, a character is received at Xbee of the bot, an interrupt is generated. Each and every character will generate an interrupt at freeRTOS, which needs to be handled. With the GUI, we are providing 3 main features. The protocol should be such that it can effectively differentiate between the 3 types of commands, and do the appropriate tasks as per the user wishes. Besides, we have to take care that each character received at Xbee will generate an interrupt.

The protocol works by collecting each character, until a special symbol is received. We have used three different symbols for the 3 commands that are controlled by the GUI. As soon as the special character is received, the buffer collecting the symbols is parsed for the correct numeric values and those numeric values are used. Finally, the buffer is cleared, to wait for new command from the GUI. As the special character for each command from the GUI is different, we are able to identify what command was actually sent from GUI.

5.3 FreeRTOS based GreenHouse Manager

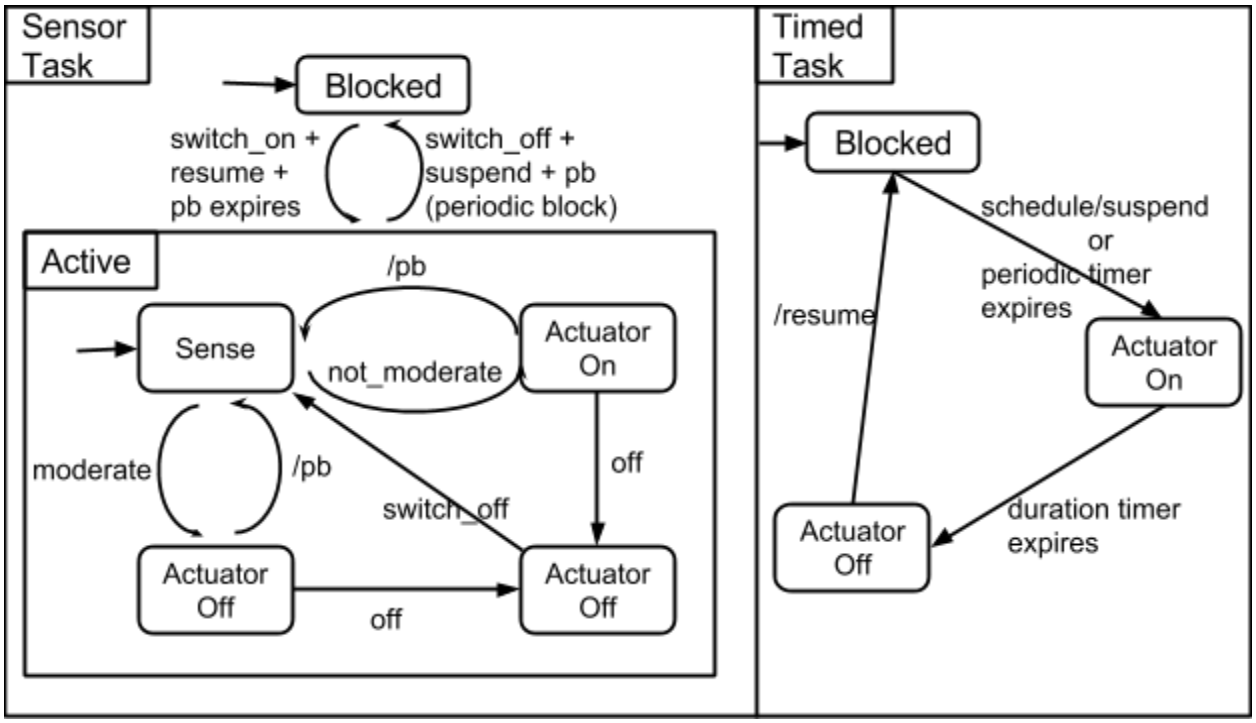
Greenhouse monitor and control system are used to maintain the sustainable environment in the Greenhouse for the crops. The environment inside Greenhouse is first sensed by installing various sensors in it. Readings of these sensors are then sent to a central monitor and control system which runs on a FreeRTOS. Based on the readings, system then sends signal to the actuators to control the environment by bringing it to desired condition. For instance,

1. Control Temperature: Temperature sensors are installed at regular spatial separation in the greenhouse. According to the readings provided by these sensors, cooler is switched on/off or its rate of cooling is changed. If temperature sensor sense high temperature, controller starts cooler to bring the temperature down.
2. Control Light: Photovoltaic sensors are installed in the greenhouse. During low light time LEDs are lit up. Series of such LEDs are hanged over crop racks/troughs. Intensity of LEDs can be increased or decreased until desired intensity level is achieved.



Our central monitor and control system consider all the sensors and actuators installed at a trough as a set and many such set of sensors and actuators can be installed at different troughs. Each set of sensor has 1)temperature sensor(LM35) 2)light sensor(LDR) 3)water pump 4)fan and 5)LED.

For each set of sensor we have certain number of tasks. See the figure above. Each set has as many sensor tasks as number of sensor. From above, we have two sensors (LM35 and LDR), therefore, two sensor task in each set. Additionally, each set has as many as timed task as number of actuators. From above, we have three actuators(fan, LED and water pump). Water pump is the actuator which don't have any corresponding sensor.



5.3.1 Sensor Task

Main work of Sensor task is to read the sensor values and react accordingly. The reaction or response trigger by the sensed reading would result in activation of one of the actuators. When the sensor task is in active state, it start from sensing state(see above statechart). It sense the readings from the sensors. If the sensor values are moderate, the task will turn off the actuator if running(goes to Actuator off state). If the sensor values are not moderate, then the task will turn on the actuators(goes to Actuator on state). After turning on/off the actuators, task again sense the sensor values and loop goes on like this.

Before sensing again, task block itself for sometime. This action is represented by periodic block(pb in above statechart). When pb timer expires it again starts from sense state.

Initially, when GreenHouse manager boots all the sensor task are blocked. User have to switch_on the sensor task. This is because if only some part of greenhouse is cultivating then we would like to sense only those parts of the greenhouse. Similarly, if some part of greenhouse is to closed down, user can switch_off the actuator. Before switching off, the sensor task first turn off the actuators if running.

5.3.2 Timed Task:

These tasks are mainly used for two purposes each of which is described below:

1. Manual override: Say, user wants to run the fan for time t, even the conditions are normal, user can specify the time and greenhouse manager would run the fan for that amount of time.
2. Actuator without sensors: Water pump has no associated sensor. User can periodically on the water pump say, every morning for 10 mins. Additionally, on a hot day, user would want to water during noon. For that user can specify the time to water and click start to water instantaneously. However, watering in noon wouldn't remembered and user have to do it again on the next day if that also is hot.

So briefly, timed task runs only when user want it to run and for the specified time. However, for the actuators with no sensors, it additionally runs periodically.

Initially, task is blocked. When user schedule the manual override and specify time duration for which actuator is to be run, timed task first blocks the corresponding sensor task. For instance, user schedule fan to run for 5 mins, task would block the sensor task which sense temperature, run the fan for 5 mins and then resume it. Thereafter, task block itself again until further scheduling. Apart from user scheduling the task, it can periodically run itself. When periodic timer expires, actuator is run for specified time and task is blocked again. This facility of periodically running the actuator is only for those actuators which don't have corresponding sensors.

An integrated Liquid crystal display (LCD) is also used for real time display of data acquired from the various sensors and the status of the various devices.

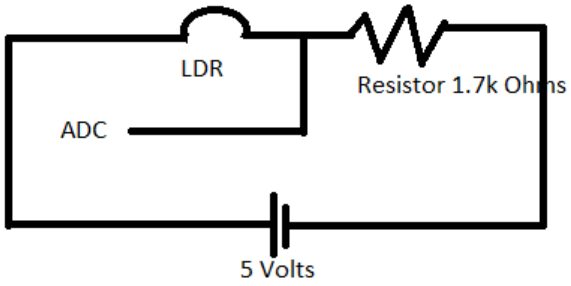
5.3.3 Implementation:

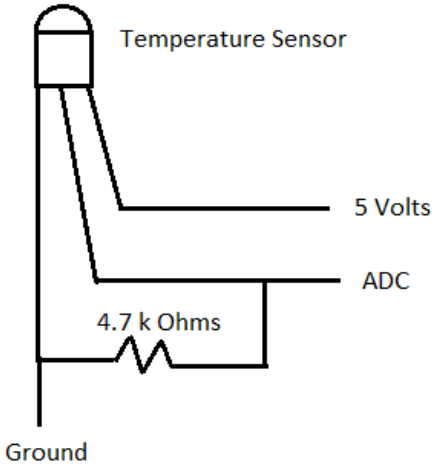
Sensor task and timed task are implemented using semaphores to avoid busy waits. Sensor task is initially blocked on a semaphore. When user switch_on the set, sensor task gets unblocked and when user again switch_off, it is again blocked on same semaphore. Similarly, timed task is blocked on a semaphore and schedule command releases it. After completion of work timed task again gets blocked on the same semaphore.

Thresholds are global variables and polled by sensor task each time while checking the moderate conditions. Time of which actuators are turned on in timed task is also global variable. When user sends the thresholds and timings from user interface, it causes interrupt. Each time new value is send, an interrupt is causes and this new value is set in the global variable.

5.4 Greenhouse prototype: Sensors and Actuators

We have cuboid of 36'' x 12'' x 60'' greenhouse where the trough with plants growing in it is kept. This cuboid is covered with plastic sheet to give it the greenhouse effect. On one wall of greenhouse we have fan which maintains the temperature of the greenhouse. On the ceiling we have LEDs to maintain proper light intensity. The circuit diagrams for light and temperature sensors are as shown in the figure.





5.5 Test Results

We have done exhaustive testing of our system. To check the correctness of our LM35 temperature sensor, we have used our soldering gun to vary the temperature. We saw that the reading of the temperature changed whenever the soldering gun was brought close to the LM35. We also tried changing the thresholds to test whether the threshold part of the code is working properly.
Regarding the LDRs, we tested the system during day and night to check whether the LDRs and their corresponding code is working accurately. Regarding scalability, we tried adding another sensor set to the

circuit and made their corresponding initializations in the code, and the second sensor set was also working correctly.
Our system passed all the tests, and the code did not run into a bug under all environments that we tested.

6. Discussion of System

We made quite a few changes from our initial plan in the SRS. Initially, we had decided to create a reusable library, which can be used as an API for future users to develop their own greenhouse projects using the library functions that we have provided. We would abstract the FreeRTOS side of the code in our library, and the functions provided in the library will be simple, easy-to-use C functions. But, during the prototype demo, the course instructor guided us to create a desktop controller instead of a library, so that the project becomes a complete application.
Hence, we shifted our focus from the FreeRTOS code to the desktop application which was developed on Java using Netbeans. Currently, we have a complete Java desktop application, which communicates with the Firebird using XBee radios, and in effect, controls the environment in the greenhouse.
Once the problem statement was changes, we thought of making just 1 sensor and actuator set, containing 2 sensors, one for light and one for temperature, and 3 actuators, one for light, one for temperature and one for irrigation. But, later on, we felt that we can improve the code by introducing scalability. So, we made a lot of abstractions in the code, and finally, we developed a completely scalable code, which can be used to create a system with as many sensor sets as we needed. All we need to do is set the GPIO and ADC pins and ports that will be used by each sensor set, and we are good to go!

7. Future Work

The code written by us is re-usable on any other robot with some changes in configurations like GPIO pins etc. The future work that can be done to extend our project is as follows :-

1. Development as a library - The current version of the code runs on LPC2148 as a standalone. We can build this into a library API which takes the pins as input and builds an executable which is able to run on various other architectures as well. An API will also help another user make use of our code to save time and efforts.
2. Two-sided thresholds - Currently our greenhouse manager can respond to only one side of the threshold. i.e. presently it acts only when the temperature is larger than the threshold, and it switches on the fan. It does not do anything if the temperature has gone too low.
3. Testing of Scalability - To test scalability, we need multiple sensor sets. Currently, we have tested for only two sensor sets. This is due to the bottleneck of the number of GPIO pins available on the Firebird 5. Having a dedicated hardware would ideally be required for deployment to a greenhouse.

8. Conclusions

This project was a great learning experience for us because this is the first time any team has developed an application on FreeRTOS. The project that has been finally implemented consists of two parts, one desktop application which communicates with the Firebird bot telling it to change thresholds, and scheduling timed tasks for a scheduled duration. The other part is the Firebird bot which contains the C code burnt on it and controlling the sensors and actuators using ADC and GPIO pins. This will effect the changes made by the controller on the greenhouse, thus allowing the greenhouse controller to manage the greenhouse environment.
In addition to the project code and documentation, we have also made a project video where we have rigorously demonstrated the working of our system, and a screen cast video with a spoken tutorial where we demonstrate how to install, configure and burn the code on the Firebird bot. The link to the Youtube video containing the screen cast is http://youtu.be/5XaSsgBbr0M.

9. References

- JSSC https://code.google.com/p/java-simple-serial-connector/w/list
- SWING http://docs.oracle.com/javase/7/docs/technotes/guides/swing/
- FreeRTOS http://www.freertos.org/tutorial/
- K. Rangan, T. Vigneswaran, “An Embedded Systems Approach to Monitor Green House”, 2010 IEEE
- Kiran Sahu, Susmita Ghosh Mazumdar, “Digitally Greenhouse Monitoring and Controlling of System based on Embedded System”, 2012 IJSER
- http://www.keil.com/uvision/
- http://www.freertos.org/FreeRTOS-quick-start-guide.html
- http://en.wikipedia.org/wiki/FreeRTOS
- http://www.facstaff.bucknell.edu/mastascu/elessonshtml/Sensors/TempLM35.html
- http://en.wikipedia.org/wiki/Photoresistor